APSP Conjecture and Boolean Matrix Multiplication

구재현

December 20, 2022

Introduction

All-Pair Shortest Path Problem

Given a **directed** graph with **nonnegative integer** weights, compute the distance between all pairs of vertices. This problem is called as an All-Pair Shortest Path (APSP).

All-Pair Shortest Path Problem

Given a **directed** graph with **nonnegative integer** weights, compute the distance between all pairs of vertices. This problem is called as an All-Pair Shortest Path (APSP).

Floyd-Warshall: Gives $O(n^3)$ algorithm.

Dijkstra: Gives $O(nm + n^2 \log n)$ algorithm.

All-Pair Shortest Path Problem

Given a **directed** graph with **nonnegative integer** weights, compute the distance between all pairs of vertices. This problem is called as an All-Pair Shortest Path (APSP).

Floyd-Warshall: Gives $O(n^3)$ algorithm.

Dijkstra: Gives $O(nm + n^2 \log n)$ algorithm.

SOTA (CW21) gives $O(\frac{n^3}{2^{\Omega(\sqrt{\log n})}})$ (If someone can pull out that $\sqrt{}$ term!)

All-Pair Shortest Path Problem

Given a **directed** graph with **nonnegative integer** weights, compute the distance between all pairs of vertices. This problem is called as an All-Pair Shortest Path (APSP).

Floyd-Warshall: Gives $O(n^3)$ algorithm.

Dijkstra: Gives $O(nm + n^2 \log n)$ algorithm.

SOTA (CW21) gives $O(\frac{n^3}{2^{\Omega(\sqrt{\log n})}})$ (If someone can pull out that $\sqrt{}$ term!)

APSP Conjecture

There is no subcubic algorithm for APSP. (An algorithm is **subcubic** if it works in $O(n^{3-\epsilon})$ for some $\epsilon > 0$.)

Subcubic Reduction

Definition (Subcubic reduction)

We say that there exists an **subcubic reduction** from A to B ($A \leq_{sc} B$) if there is an algorithm Z with oracle access to B such that for every $\epsilon > 0$ there is a $\delta > 0$ satisfying three properties:

- For every instance x of A, $Z(x)$ solves the problem A on x.
- Z runs in $O(n^{3-\delta})$ time.
- For every $x \in A$, let n_i be the size of i-th oracle call to B in $Z(x)$. Then $\sum_i n_i^{3-\epsilon} \leq n^{3-\delta}$.

The book's definition is **INCORRECT**! Be careful. The above definition is from the original paper defining subcubic reduction.

Subcubic Reduction

Definition (Subcubic equivalent)

A and B are subcubic equivalent ($A \equiv_{sc} B$) if $A \leq_{sc} B$ and $B \leq_{sc} A$

Definition (APSP-hard)

A is APSP-hard if $APSP \leq_{sc} A$.

Definition (APX-complete)

A is APSP-complete if A is subcubic equivalent to APSP.

Subcubic Reduction

Definition (Subcubic equivalent)

A and B are subcubic equivalent ($A \equiv_{sc} B$) if $A \leq_{sc} B$ and $B \leq_{sc} A$

Definition (APSP-hard)

A is APSP-hard if $APSP \leq_{sc} A$.

Definition (APX-complete)

A is APSP-complete if A is subcubic equivalent to APSP.

Those are all standard definitions.

APSP-hard problems are at least cubic.

APSP-complete problems are cubic.

Both are likely not subcubic.

Subcubic Hardness Must Go On

Earlier, we said the graph is a **directed** graph with **nonnegative integer** weights. Does things get easier if we relax some of these conditions?

Subcubic Hardness Must Go On

Earlier, we said the graph is a **directed** graph with **nonnegative integer** weights. Does things get easier if we relax some of these conditions?

APSP in Undirected Graph (Folklore)

Undirected APSP is APSP-complete.

Subcubic Hardness Must Go On

Earlier, we said the graph is a **directed** graph with **nonnegative integer** weights. Does things get easier if we relax some of these conditions?

APSP in Undirected Graph (Folklore)

Undirected APSP is APSP-complete.

Theorem (Zwick, 1998)

When all edge weights have **absolute value** of at most $O(1)$, APSP can be solved in $O(n^{2.529})$ time.

Theorem (Zwick, 1999)

When all edge weights have **absolute value** of at most $O(1)$, Undirected APSP can be solved in $O(n^{\omega})$ time ($\omega < 2.373$).

Theorem (Zwick, 1998)

APSP can be $1+\epsilon$ approximated in $\tilde{O}(\frac{n^{\omega}}{\epsilon})$ time.

Subcubic??? Hardness Must Go On

The discrepancy between the undirected and directed cases is intriguing and there are attempts to understand this gap.

If all falls down...

Subcubic??? Hardness Must Go On

The discrepancy between the undirected and directed cases is intriguing and there are attempts to understand this gap.

If all falls down...

Problem (All-Edges Monochromatic Triangle)

Given a graph with n vertices, and edges labeled with integer colors, decide for each edge if it belongs to a monochromatic triangle.

Problem (Min-Max Product)

Given two matrices A, B, compute a matrix C where $C_{i,j} = \min_k \max(A_{i,k}, B_{k,j})$.

Subcubic??? Hardness Must Go On

Theorem (LPV20)

If All-Edges Monochromatic Triangle has a $T(n)$ algorithm, then the unweighted APSP can be solve d in $T(n)\log^2 n$ time.

Theorem (LPV20)

If Min-Max Product has a $T(n)$ algorithm, then the unweighted APSP can be solved in $T(n)\log n$ time.

Theorem

Both problems can be solved in $\tilde{O}(n^{\frac{\omega+1}{2}})$ time.

We can do this for all figures. Imagine a class of $\tilde{O}(n^{\omega}), \tilde{O}(n^{(\omega+2)/3}), \tilde{O}(n^{(\omega+1)/2})$, on and on...

Boolean Matrix Multiplication Conjecture

This conjecture is only shortly mentioned in the book.

I think it is of great importance due to its impact on various problems (especially CP-style ones), and the controversial aspect.

Boolean Matrix Multiplication Conjecture

This conjecture is only shortly mentioned in the book.

I think it is of great importance due to its impact on various problems (especially CP-style ones), and the controversial aspect.

Falsified BMM Conjecture

There is no subcubic algorithm to multiply two boolean matrices (where the addition is $\vee$ and multiplication is $\wedge$).

Boolean Matrix Multiplication Conjecture

This conjecture is only shortly mentioned in the book.

I think it is of great importance due to its impact on various problems (especially CP-style ones), and the controversial aspect.

Falsified BMM Conjecture

There is no subcubic algorithm to multiply two boolean matrices (where the addition is $\vee$ and multiplication is $\wedge$).

But this does not work. You can very easily reduce this into an integer and use $O(n^\omega)$ matrix multiplication.

We use the notion of **combinatorial** algorithm.

Boolean Matrix Multiplication Conjecture

BMM Conjecture

There is no **combinatorial** subcubic algorithm to multiply two boolean matrices (where the addition is $\vee$ and multiplication is $\wedge$).

Boolean Matrix Multiplication Conjecture

BMM Conjecture

There is no **combinatorial** subcubic algorithm to multiply two boolean matrices (where the addition is $\vee$ and multiplication is $\wedge$).

So what is the definition of *combinatorial* algorithm? Well...

> The notion of a "combinatorial" algorithm does not have a formal definition. Intuitively, such algorithms are not only theoretically but also practically efficient. In the above theorem, by combinatorial we mean an algorithm with low leading constants. One can verify that the reductions in our article have low leading constants and low overhead; hence, any simple fast triangle algorithm would yield a simple (and only slightly slower) BMM algorithm. The relation between BMM and the triangle problem has been investigated by many researchers, e.g., Reference [73] (Open Problem 4.3(c)) and Reference [57] (Open Problem 8.1).

Boolean Matrix Multiplication Conjecture

BMM Conjecture

There is no **combinatorial** subcubic algorithm to multiply two boolean matrices (where the addition is $\vee$ and multiplication is $\wedge$).

So what is the definition of *combinatorial* algorithm? Well...

> The notion of a "combinatorial" algorithm does not have a formal definition. Intuitively, such algorithms are not only theoretically but also practically efficient. In the above theorem, by combinatorial we mean an algorithm with low leading constants. One can verify that the reductions in our article have low leading constants and low overhead; hence, any simple fast triangle algorithm would yield a simple (and only slightly slower) BMM algorithm. The relation between BMM and the triangle problem has been investigated by many researchers, e.g., Reference [73] (Open Problem 4.3(c)) and Reference [57] (Open Problem 8.1).

Basically, an algorithm is **combinatorial** if it's not Strassen-like, because they are, well, pretty bad.

Boolean Matrix Multiplication Conjecture

I guess that the notion of BMM arises from the following sentiment:

1. Most subcubic achievement in the field are done by reduction to the Matrix Multiplication. They are practically useless, and we know that they never will be.
2. If we assume BMM to be hard, we obtain stronger tool to obtain subcubic hardness, and stop wasting time crunching numbers.
3. Conjectures can appeal to feelings. :)

Boolean Matrix Multiplication Conjecture

I guess that the notion of BMM arises from the following sentiment:

1. Most subcubic achievement in the field are done by reduction to the Matrix Multiplication. They are practically useless, and we know that they never will be.
2. If we assume BMM to be hard, we obtain stronger tool to obtain subcubic hardness, and stop wasting time crunching numbers.
3. Conjectures can appeal to feelings. :)

On the other hand, If you don't buy the *combinatorical* assumption, BMM Conjecture is still effective - anyway, we are not expecting almost quadratic BMM algorithm anytime soon.

Boolean Matrix Multiplication Conjecture

BMM Conjecture is orthogonal to APSP Conjecture.

1. Falsification of APSP do not falsify BMM as the algorithm can be non-combinatorial.
2. Falsification of BMM do not falsify APSP as you can't solve APSP with BMM.

But they are still closely related.

Boolean Matrix Multiplication Conjecture

BMM Conjecture is orthogonal to APSP Conjecture.

1. Falsification of APSP do not falsify BMM as the algorithm can be non-combinatorial.
2. Falsification of BMM do not falsify APSP as you can't solve APSP with BMM.

But they are still closely related.

There is a combinatorial $O(n^3 \text{poly}(\log\log n)/\log^4 n)$ algorithm for BMM using Four Russians technique (Huachang Yu, 2015)

Boolean Matrix Multiplication Conjecture

BMM Conjecture is orthogonal to APSP Conjecture.

1. Falsification of APSP do not falsify BMM as the algorithm can be non-combinatorial.
2. Falsification of BMM do not falsify APSP as you can't solve APSP with BMM.

But they are still closely related.

There is a combinatorial $O(n^3\text{poly}(\log\log n)/\log^4 n)$ algorithm for BMM using Four Russians technique (Huachang Yu, 2015)

But why is it a combinatorial algorithm? Because it doesn't falsify BMM? XD

Debate

One reason why mathematics enjoys special esteem, above all other sciences, is that its laws are absolutely certain and indisputable, while those of other sciences are to some extent debatable and in constant danger of being overthrown by newly discovered facts - Albert Einstein, 1921

Debate

One reason why mathematics enjoys special esteem, above all other sciences, is that its laws are absolutely certain and indisputable, while those of other sciences are to some extent debatable and in constant danger of being overthrown by newly discovered facts - Albert Einstein, 1921

Mathematics exists to model real life. Complexity analysis is an abstraction to real-life machine runtime. Is it correct to do maths which is about exploiting the weakness of complexity analysis and no else?

If a conjecture is falsified, it should resort to something. But this conjecture appeals to *feelings*, at best practical implementation attempts. If BMM Conjecture is attacked you can also accuse them of being *non-combinatorial*. Is it mathematics or politics?

Debate

One reason why mathematics enjoys special esteem, above all other sciences, is that its laws are absolutely certain and indisputable, while those of other sciences are to some extent debatable and in constant danger of being overthrown by newly discovered facts - Albert Einstein, 1921

Mathematics exists to model real life. Complexity analysis is an abstraction to real-life machine runtime. Is it correct to do maths which is about exploiting the weakness of complexity analysis and no else?

If a conjecture is falsified, it should resort to something. But this conjecture appeals to *feelings*, at best practical implementation attempts. If BMM Conjecture is attacked you can also accuse them of being *non-combinatorial*. Is it mathematics or politics?

I think this is a philosophical question!

Application

Problem known to be APSP-Complete

THEOREM 1.1. *The following problems (with weights in* $\{-M, \ldots, M\} \cup \{-\infty, \infty\}$*) either* all *have truly subcubic algorithms, or* none *of them do:*

(1) The all-pairs shortest paths problem on weighted digraphs (APSP).
(2) The all-pairs shortest paths problem on undirected weighted graphs.
(3) Detecting if a weighted graph has a triangle of negative total edge weight.
(4) Listing up to $n^{3-\delta}$ *negative triangles in an edge-weighted graph, for a fixed* $\delta > 0$.
(5) Computing the matrix product over the (min, +)*-semiring.*
(6) Verifying the correctness of a matrix product over the (min, +)*-semiring.*
(7) Checking whether a given matrix defines a metric.
(8) Finding a minimum weight cycle in a graph of non-negative edge weights.
(9) The replacement paths problem on weighted digraphs.
(10) Finding the second *shortest simple path between two nodes in a weighted digraph.*
(11) Finding a maximum subarray of a given matrix.

The book also mentions the following:

1. Radius: $\min_{s \in V} \max_{t \in V} dist(s, t)$ and argmin resp.
2. Median: $\min_{s \in V} \sum_{t \in V} dist(s, t)$ and argmin resp.,
3. Betweenness Centrality: For all $s, t \in V - \{v\}$, compute the sum fraction of shortest path between s, t that contains v
4. Check if the given matrix is the correct APSP distance matrix.

The Diameter Problem

The **diameter** of graph is the longest shortest path between two vertices s, t.

Clearly, Diameter is easier than APSP. It is not known whether the Diameter is APSP-Complete.

The following problems are subcubic equivalent to Diameter (All excerpts from the book.)

1. Positive Betweenness Centrality: Determine if the Betweenness Centrality of v is positive.
2. Reach Centrality of v:
   $\max_{s,t \in V, dist(s,t)=dist(s,v)+dist(v,t)} \min(dist(s,v), dist(v,t))$

PosBetCent is harder than Diameter

We first prove Diameter $\leq_{sc}$ PosBetCent. Remind that the problem PosBetCent is about determining if there exists a shortest path passing through a specified vertex.

PosBetCent is harder than Diameter

We first prove Diameter $\leq_{sc}$ PosBetCent. Remind that the problem PosBetCent is about determining if there exists a shortest path passing through a specified vertex.

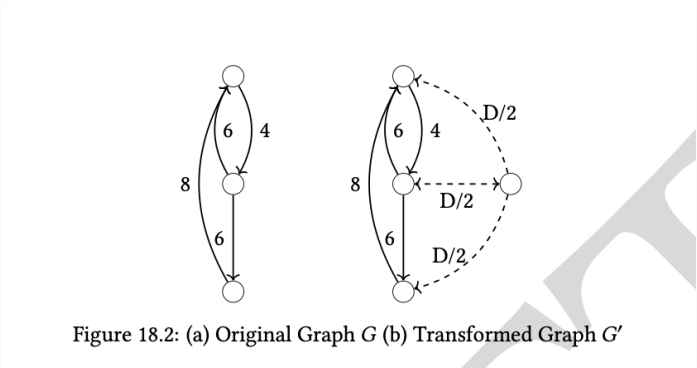

Figure 18.2: (a) Original Graph G (b) Transformed Graph G'

Suppose we have a solver for PosBetCent. We add a new vertex x and connect it to all other vertices with weight D.

PosBetCent is harder than Diameter

We first prove Diameter $\leq_{sc}$ PosBetCent. Remind that the problem PosBetCent is about determining if there exists a shortest path passing through a specified vertex.

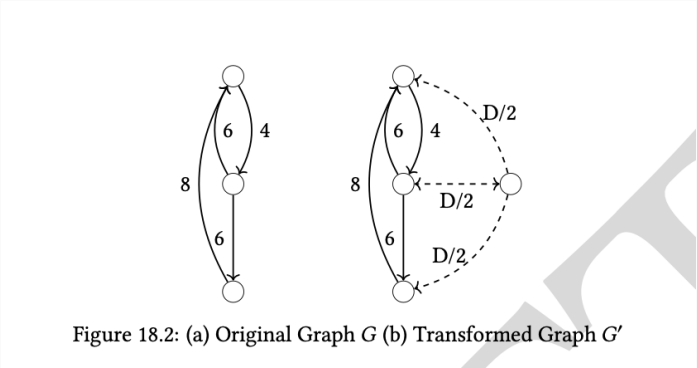

Figure 18.2: (a) Original Graph G (b) Transformed Graph G'

Suppose we have a solver for PosBetCent. We add a new vertex x and connect it to all other vertices with weight D.

- If D is large, PosBetCent will be false.
- If D is small, PosBetCent will be true.
- Demarcation point is exactly half of the diameter.
- Binary search suffices!

Diameter is harder than PosBetCent

We prove PosBetCent $\leq_{sc}$ Diameter.

Remember, we want to find some $s, t \in V - \{x\}$ such that $dist(s,t) = dist(s,x) + dist(x,t)$.

Diameter is harder than PosBetCent

We prove PosBetCent $\leq_{sc}$ Diameter.

Remember, we want to find some $s, t \in V - \{x\}$ such that $dist(s,t) = dist(s,x) + dist(x,t)$.

Let $M = \max_{e \in E} w_e$ and $U = 3M|V|$.

For each vertex $v \in V$ we compute $dist(v,x)$ and $dist(x,v)$. This can be done with two iterations of Dijkstra's algorithm which is quadratic.

We will construct a graph, where its diameter can be used to decide if PosBetCent(x) is true.

Diameter is harder than PosBetCent

The graph has $3|V| - 2$ vertices:
$V \cup \{v_a | v \in V - \{x\}\} \cup \{v_b | v \in V - \{x\}\}$.

The graph has $|E| + 2|V| - 2$ edges:

1. Original edges.
2. For $v \in V - \{x\}$, connect $v_a \to v$ with weight $U - dist(v, x)$.
3. For $v \in V - \{x\}$, connect $v \to v_b$ with weight $U - dist(x, v)$.
4. For $v \in V$, connect $v \to v_a$ with weight 0.
5. For $v \in V$, connect $v_b \to v$ with weight 0.

Diameter is harder than PosBetCent

The graph has $3|V| - 2$ vertices:
$V \cup \{v_a | v \in V - \{x\}\} \cup \{v_b | v \in V - \{x\}\}$.

The graph has $|E| + 2|V| - 2$ edges:

1. Original edges.
2. For $v \in V - \{x\}$, connect $v_a \to v$ with weight $U - dist(v, x)$.
3. For $v \in V - \{x\}$, connect $v \to v_b$ with weight $U - dist(x, v)$.
4. For $v \in V$, connect $v \to v_a$ with weight 0.
5. For $v \in V$, connect $v_b \to v$ with weight 0.

Observe that the shortest path should start in some v_a and end in some v_b. Let $s_a \to s \to t \to t_b$ be the longest path. The distance should be
$2U - dist(s, x) - dist(x, t) + dist(s, t) =$
$2U - (dist(s, x) + dist(x, t) - dist(s, t))$.

Diameter is harder than PosBetCent

The graph has $3|V| - 2$ vertices:
$V \cup \{v_a | v \in V - \{x\}\} \cup \{v_b | v \in V - \{x\}\}$.

The graph has $|E| + 2|V| - 2$ edges:

1. Original edges.
2. For $v \in V - \{x\}$, connect $v_a \to v$ with weight $U - dist(v, x)$.
3. For $v \in V - \{x\}$, connect $v \to v_b$ with weight $U - dist(x, v)$.
4. For $v \in V$, connect $v \to v_a$ with weight 0.
5. For $v \in V$, connect $v_b \to v$ with weight 0.

Observe that the shortest path should start in some v_a and end in some v_b. Let $s_a \to s \to t \to t_b$ be the longest path. The distance should be
$2U - dist(s, x) - dist(x, t) + dist(s, t) =$
$2U - (dist(s, x) + dist(x, t) - dist(s, t))$.

This quantity is at most $2U$ due to triangle inequality. If the equality holds, then PosBetCent(x) is true.

Problem known to be BMM-Complete or BMM-hard

THEOREM 1.3. *Either all of the following have truly subcubic "combinatorial" algorithms, or none of them:*

- *Boolean matrix multiplication (BMM).*
- *Detecting if a graph has a triangle.*
- *Listing up to $n^{3-\delta}$ triangles in a graph for constant $\delta > 0$.*
- *Verifying the correctness of a matrix product over the Boolean semiring.*

Problem known to be BMM-Complete or BMM-hard

THEOREM 1.3. *Either all of the following have truly subcubic "combinatorial" algorithms, or none of them:*

- *Boolean matrix multiplication (BMM).*
- *Detecting if a graph has a triangle.*
- *Listing up to $n^{3-\delta}$ triangles in a graph for constant $\delta > 0$.*
- *Verifying the correctness of a matrix product over the Boolean semiring.*

Assuming the BMM, we can prove that no combinatorial algorithm can respond the *range mode (most frequent element) query* in $o(\sqrt{n})$ time. You can find proofs in Aeren's secmem posts.

Assuming the BMM. we can prove that no combinatorial algorithm can solve all-pair reachability problem in $o(n^3)$ time. You can find proofs in koosaga's secmem posts.

Triangle Detection is subcubic equivalent to BMM

If we can solve BMM in subcubic time, you can simply look up the entries of A^3 to find the triangle.

The opposite direction is nontrivial. We can prove the following theorem:

Theorem

Suppose that a triangle in an n node graph can be detected in $D(n)$ time. Then BMM of two $n \times n$ matrices can be computed in $O(n^2 D(n^{1/3}))$ time.

Let's prove this!

Triangle Detection is subcubic equivalent to BMM

If we can check if the triangle exists in $T(n)$ (where $T(n)/n$ is nondecreasing), we can always find an actual triangle certificate in $O(T(n))$.

Triangle Detection is subcubic equivalent to BMM

If we can check if the triangle exists in $T(n)$ (where $T(n)/n$ is nondecreasing), we can always find an actual triangle certificate in $O(T(n))$.

Make two new duplicates of vertices to turn the graph into tripartite.

Halve each of three vertex sets and try for all 2^3 combinations to check which half contains a 3-cycle. Recurse down to the half which contains a 3-cycle.

We have $T'(n) = 8T(n) + T'(n/2) = O(T(n))$.

Triangle Detection is subcubic equivalent to BMM

Consider a graph G' with $3n$ vertices v_i, v_j, v_k $(1 \leq v \leq n)$. If $A[u, v]$, add an edge between u_i, v_j. If $B[u, v]$, add an edge between u_j, v_k.

Model BMM problem as the following: If there is an edge between (u_i, v_j) and (v_j, w_k), add an edge between (u_i, w_k).

Idea: We can invert the edge set between $I \times K$, find the triangles, and remove the edge in $I \times K$.

Triangle Detection is subcubic equivalent to BMM

Consider a graph G' with $3n$ vertices v_i, v_j, v_k $(1 \leq v \leq n)$. If $A[u, v]$, add an edge between u_i, v_j. If $B[u, v]$, add an edge between u_j, v_k.

Model BMM problem as the following: If there is an edge between (u_i, v_j) and (v_j, w_k), add an edge between (u_i, w_k).

Idea: We can invert the edge set between $I \times K$, find the triangles, and remove the edge in $I \times K$.

Split each tripartition into t pieces, each on n/t nodes. For all t^3 triple of pieces (I_i, J_j, K_k), find the triangle, and delete it. How fast is it?

Triangle Detection is subcubic equivalent to BMM

Consider a graph G' with $3n$ vertices v_i, v_j, v_k $(1 \leq v \leq n)$. If $A[u, v]$, add an edge between u_i, v_j. If $B[u, v]$, add an edge between u_j, v_k.

Model BMM problem as the following: If there is an edge between (u_i, v_j) and (v_j, w_k), add an edge between (u_i, w_k).

Idea: We can invert the edge set between $I \times K$, find the triangles, and remove the edge in $I \times K$.

Split each tripartition into t pieces, each on n/t nodes. For all t^3 triple of pieces (I_i, J_j, K_k), find the triangle, and delete it. How fast is it?

1. For each instance, we use $1 +$ (actually deleted triangle count) number of triangle finding calls.
2. The 1 sums to the number of instances, which is t^3.
3. The deleted triangle count sums to n^2.
4. We have $O((n^2 + t^3)D(n/t))$. Setting $t = n^{2/3}$, we obtain $O(n^2 D(n^{1/3}))$ time.

OR of the conjectures

When we assume that **at least one** of APSP Conjecture, SETH, 3SUM Conjecture is true, we obtain the following result on the dynamic graph problems:

1. Directed graphs, edge updates, number of SCC.
2. Directed graphs, one source node, edge updates, number of nodes reachable from s.
3. Undirected graphs, vertex update with one source node, vertex turn on/off, number of nodes reachable from s.
4. Directed graphs with capacity in $[n]$, a source, a sink. Max flow.

You must have either amortized $n^{1-o(1)}$ update or query times, or $n^{3-o(1)}$ preprocessing times.

OR of the conjectures

When we assume that **at least one** of APSP Conjecture, SETH, 3SUM Conjecture is true, we obtain the following result on the dynamic graph problems:

1. Directed graphs, edge updates, number of SCC.
2. Directed graphs, one source node, edge updates, number of nodes reachable from s.
3. Undirected graphs, vertex update with one source node, vertex turn on/off, number of nodes reachable from s.
4. Directed graphs with capacity in $[n]$, a source, a sink. Max flow.

You must have either amortized $n^{1-o(1)}$ update or query times, or $n^{3-o(1)}$ preprocessing times.

The same paper claims the following: Assuming SETH, Single source, all sink max-flow can not be solved in $O(n^{2-\epsilon})$ time. Therefore, in directed graphs, max flow queries are best done naively.

OR of the conjectures

If we assume that **at least one** of APSP Conjecture, SETH, 3SUM Conjecture is true, we can not solve this problem in $O(n^{3-\epsilon})$ time:

Triangle-Collection

Given a node-colored graph G, is it true that for all triples of distinct colors a, b, c there is a triangle (x, y, z) in G in which x has color a, y has color b, and z has color c? (Does the set of all triangles in the graph **collect** all triples of colors?)

OR of the conjectures

If we assume that **at least one** of APSP Conjecture, SETH, 3SUM Conjecture is true, we can not solve this problem in $O(n^{3-\epsilon})$ time:

Triangle-Collection

Given a node-colored graph G, is it true that for all triples of distinct colors a, b, c there is a triangle (x, y, z) in G in which x has color a, y has color b, and z has color c? (Does the set of all triangles in the graph **collect** all triples of colors?)

We can claim that each of the conjectures implies a cubic algorithm for the above problem. The proof is pretty technical.

Then we can use the aforementioned dynamic problems to obtain an algorithm for this problem.

Further reading

Zwick, Uri. "All pairs shortest paths using bridging sets and rectangular matrix multiplication." (2002)

Williams, Virginia Vassilevska, and Ryan Williams. "Subcubic equivalences between path, matrix and triangle problems." (2010)

Abboud, Amir, Fabrizio Grandoni, and Virginia Vassilevska Williams. "Subcubic equivalences between graph centrality problems, APSP and diameter." (2014)

Abboud, Amir, Virginia Vassilevska Williams, and Huacheng Yu. "Matching triangles and basing hardness on an extremely popular conjecture." (2015)